FORNAX

Variational Inference

Krzysztof Kolasiński, 14.11.2017

Content

- Motivation
- Linear regression y=wx+b
- Variational Inference for LR
- Examples in Edward

A motivations for studying Bayesian methods

PROS:

- Bayesian framework allows for capturing model uncertainty (medicine)
- Can incorporate prior knowledge
- Useful when dealing with small dataset
- More robust against overfitting
- Possibly more robust against adversarial examples (some promising results)

CONS:

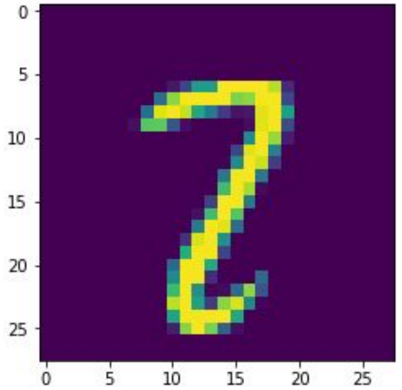
- computationally more demanding,
- in many applications we are forced to use approximations,
- in case of deep learning problem still wore in performance than standard approach (MLE)
- Can incorporate prior knowledge

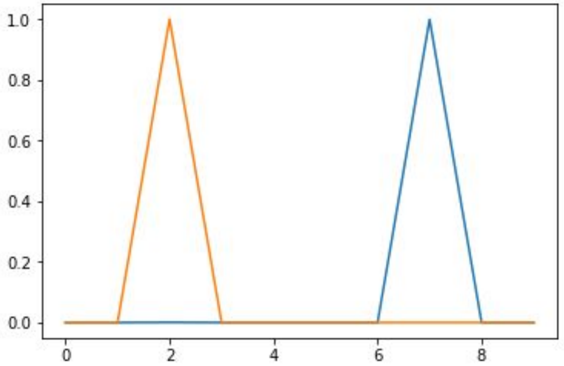
The motivation - demos

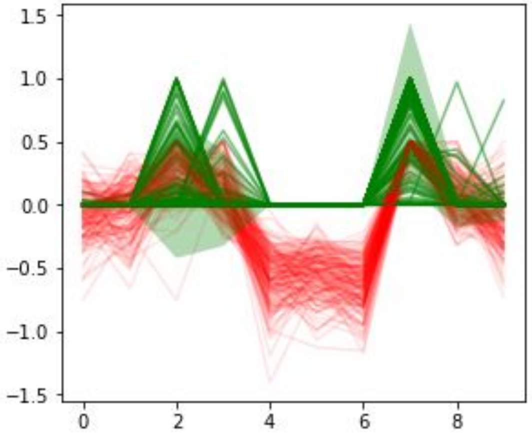
The biggest advantage of using Bayesian approach is knowledge about prediction uncertainty

We already have some experience with MCDropout

Badly classified example



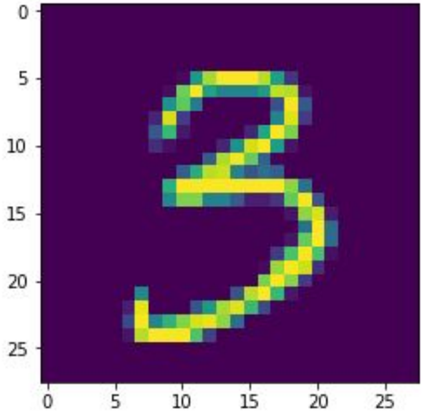


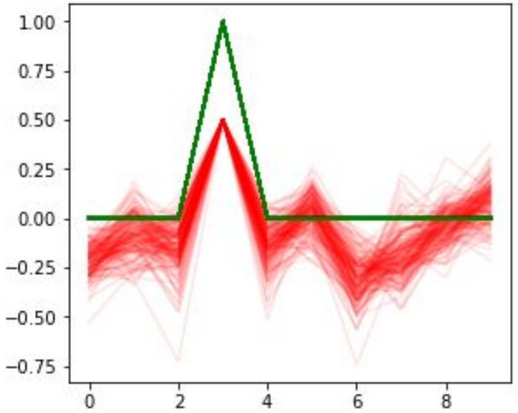


Predictions of standard model.
Dropout switched to test mode

Samples from permanent dropout - same model, same weights.

Correctly classified example



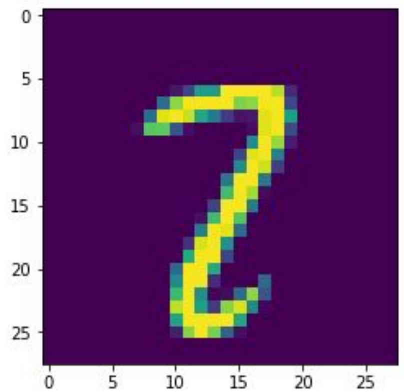


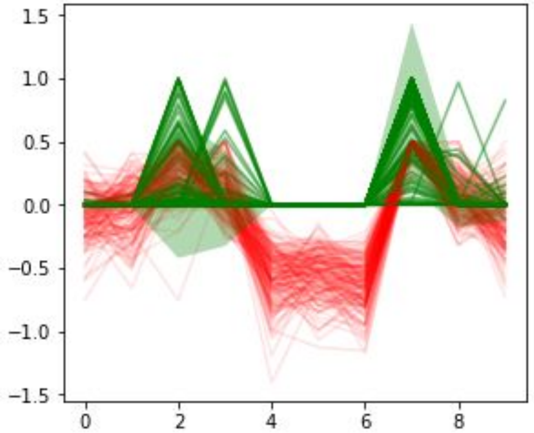
We captured the uncertainty by measuring the mean entropy of class probabilities over computed samples

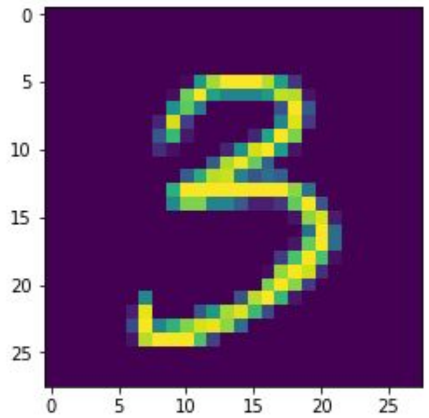
$$H_s = -\sum p_{s,i} \log\left(p_{s,i}\right) \qquad \Delta = \frac{1}{N_s} \sum_{s=1}^{N_s} H_s$$

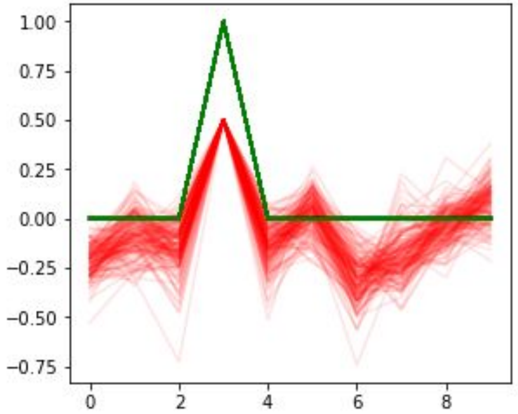
The motivation

The biggest advantage of using Bayesian approach is knowledge about prediction uncertainty









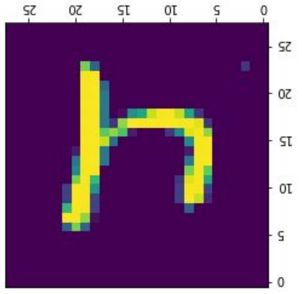
We captured the uncertainty by measuring the mean entropy of class probabilities over computed samples. With N_s~32.

$$H_s = -\sum p_{s,i} \log\left(p_{s,i}\right) \quad \Delta = \frac{1}{N_s}\sum_{s=1}^{N_s} H_s$$

On separate dataset (validation) we computed the “best” threshold value **delta**$_{th}$ above which we categorized samples as unreadable.

$$\Delta > \delta_{th}$$

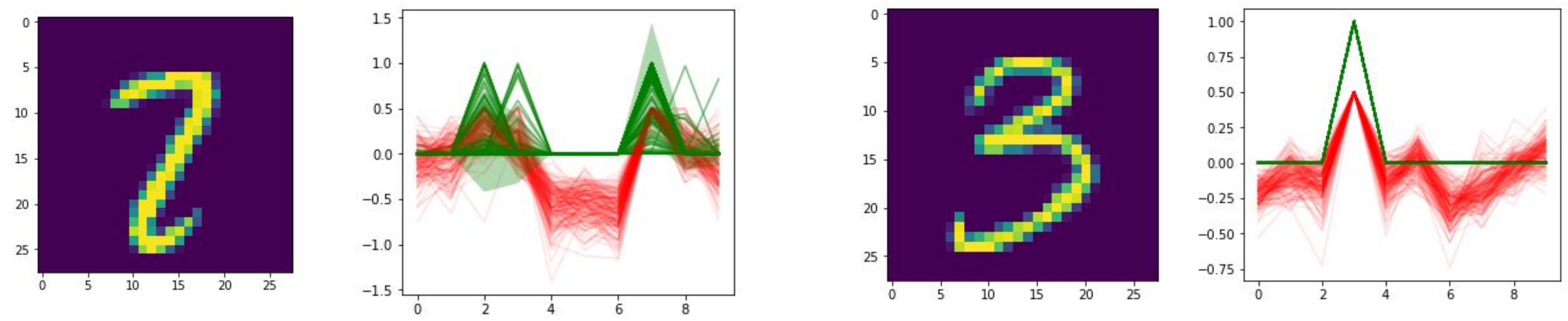
The same technique was applied to find rotated objects in detection pipeline.



- MCDropout improved our predictions
- but at cost of longer inference time since sampling is required.

Bayesian approach as alternative for MCDropout (for us)

The biggest advantage of using Bayesian approach is knowledge about prediction uncertainty



- MCDropout drops randomly activations in model,
- Bayesian networks can do it by explicitly modeling models parameters as random variables
- In Variational Inference we implement them using the reparametrization trick:

$$W = W_{mean} + \epsilon W_{sigma}$$

$$\epsilon \sim N(0,1)$$

- During the inference we sample W by sampling epsilon and then perform prediction.
- The idea is to apply Bayesian inference instead of MCDropout and check if it will work better.

A linear regression case

A linear regression case

We are going to solve the simplest regression problem: linear regression of form y = w * x + b

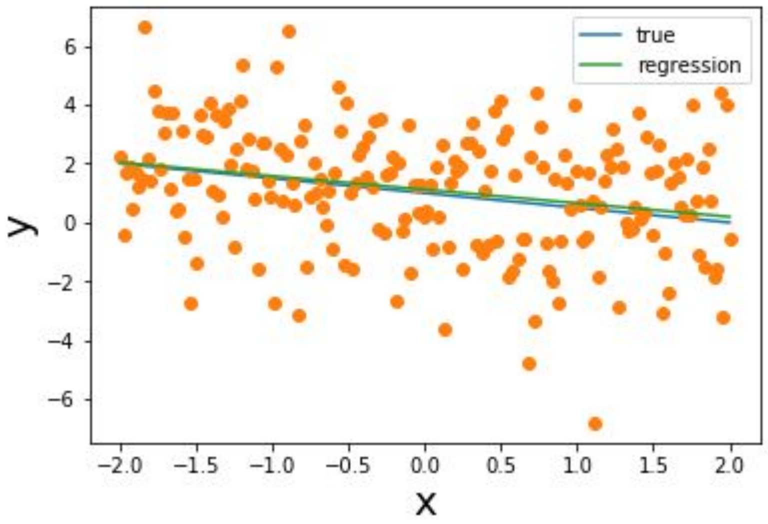
$$y = wx + b + \varepsilon$$

Here we assume that problem has linear dependency on x but data is corrupted by gaussian noise - epsilon

$$\varepsilon \propto N(0, \sigma^2)$$

Let’s define observed data: (xi, yi), with the probability of observing data pair given by

$$p(y_i|x_i, \theta) = C \exp\left(-\frac{1}{2\sigma^2}\left(y_i - f\left(x_i\right)\right)^2\right)$$



where:

$$f(x) = wx + b \qquad \theta = (w, b)$$

A linear regression case

The probability of observing data is given by

where:

$$p(y_i|x_i,\theta) = C\exp\left(-\frac{1}{2\sigma^2}\left(y_i - f\left(x_i\right)\right)^2\right)$$

$$f\left(x\right) = wx + b \qquad \theta = \left(w, b\right)$$

Assuming that samples *i* are independent the total probability is given by:

$$p(\mathbf{Y}|\mathbf{X},\theta) = C^n\exp\left(-\frac{1}{2\sigma^2}\left(\mathbf{Y} - f\left(\mathbf{X}\right)\right)^T\left(\mathbf{Y} - f\left(\mathbf{X}\right)\right)\right)$$

Let's find the MLE by maximizing the logarithm of p(Y| X)

$$\mathcal{L}_\theta = \log\left(p\right) = \log C^n - \frac{1}{2\sigma^2}\left(\mathbf{Y} - f\left(\mathbf{X}\right)\right)^T\left(\mathbf{Y} - f\left(\mathbf{X}\right)\right)$$

Remember that this is equivalent to minimizing **KL(p_data || p_model)**

A linear regression case

The optimal parameters can be found analytically one f(x) is simple enough

$$\mathcal{L}_\theta = \log(p) = \log C^n - \frac{1}{2\sigma^2}(\mathbf{Y} - f(\mathbf{X}))^T(\mathbf{Y} - f(\mathbf{X}))$$

$$f(x) = wx + b \qquad \theta = (w, b)$$

For function of form f(x) we have:

$$w = -\frac{\mathbb{E}_{xy} - \mathbb{E}_x\mathbb{E}_y}{\mathbb{E}_x^2 - \mathbb{E}_{x^2}} \qquad b = \mathbb{E}_y - w\mathbb{E}_x$$

```
w_reg = - ((x*y).mean() - x.mean() * y.mean())/((x).mean()**2 - (x**2).mean())
b_reg = y.mean() - w_reg * x.mean()
```

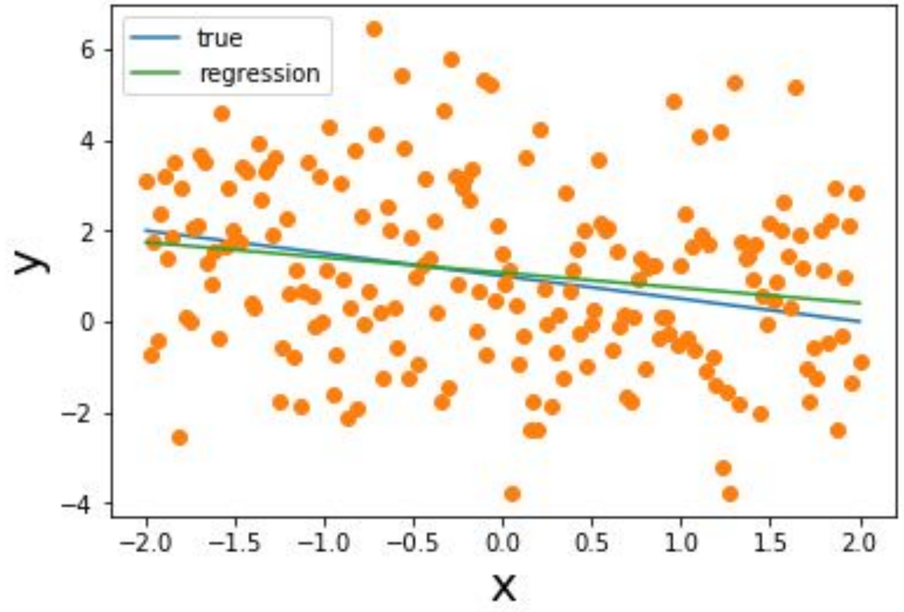
A linear regression case

We solved well known problem of linear regression

$$f(x) = wx + b$$

$$w = -\frac{\mathbb{E}_{xy} - \mathbb{E}_x \mathbb{E}_y}{\mathbb{E}_x^2 - \mathbb{E}_{x^2}}$$

$$b = \mathbb{E}_y - w\mathbb{E}_x$$

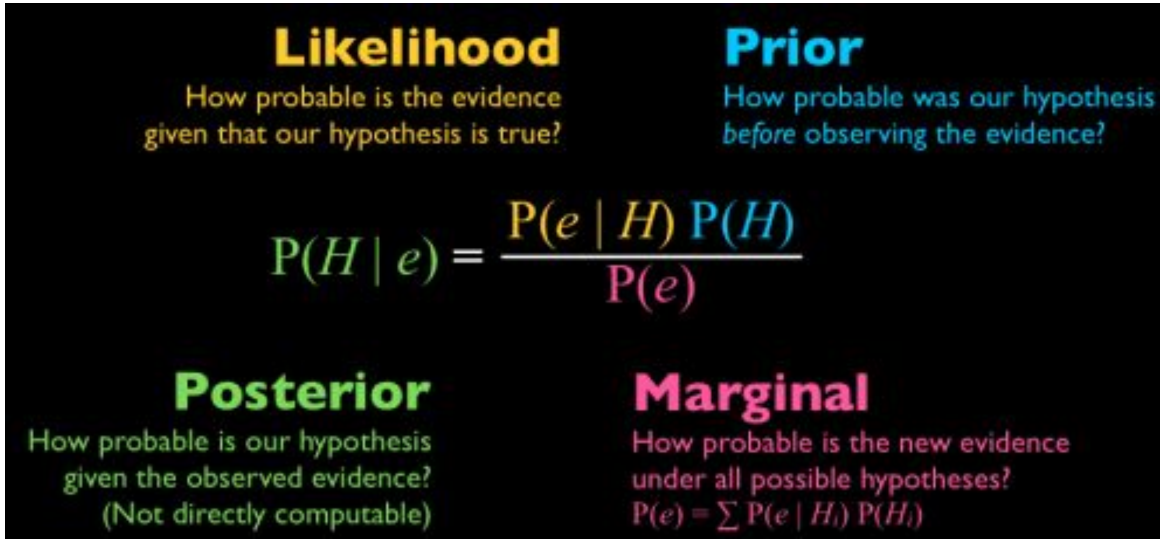


We have found optimal parameters by maximizing likelihood:

$$p(\mathbf{Y}|\mathbf{X}, \theta) = C^n \exp\left(-\frac{1}{2\sigma^2}\left(\mathbf{Y} - f(\mathbf{X})\right)^T \left(\mathbf{Y} - f(\mathbf{X})\right)\right)$$

The Bayesian rule

We want to add information about uncertainty of our model



The likelihood - the probability of observing Y given X and model parameters

$$p(\mathbf{Y}|\mathbf{X},\theta) = C^n \exp\left(-\frac{1}{2\sigma^2}\left(\mathbf{Y} - f(\mathbf{X})\right)^T\left(\mathbf{Y} - f(\mathbf{X})\right)\right)$$

The prior - our best knowledge about probability distribution of model parameters. Not trainable.

$$p(\theta) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(w^2 + b^2\right)\right)$$

The marginal - can be understand as normalization constant, similar as in softmax function, partition function etc...

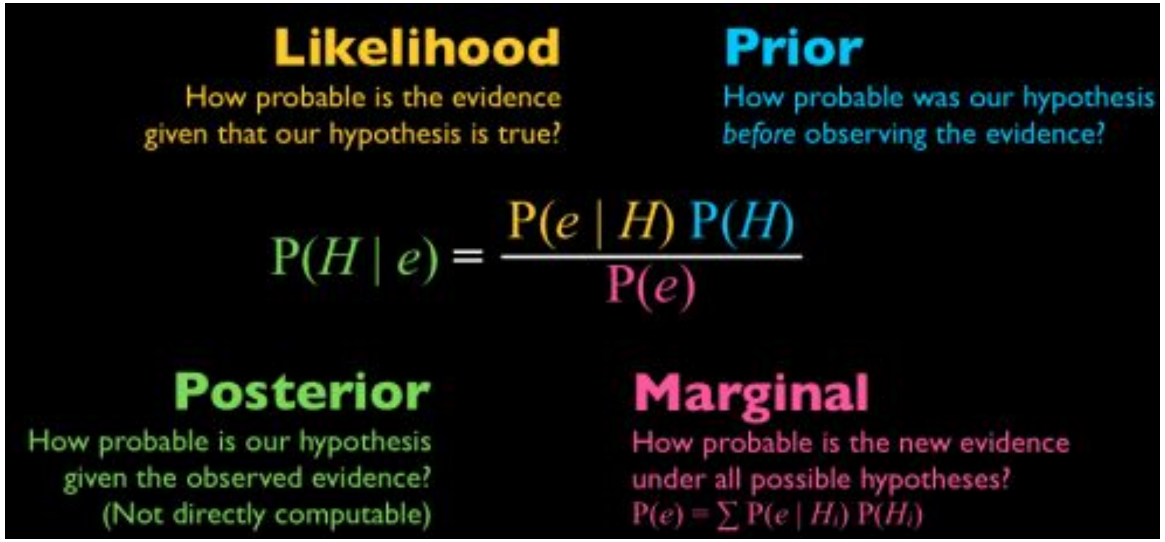
The Bayesian rule

Let’s apply BR to our problem and for the sake of clarity we omit X from following expressions:

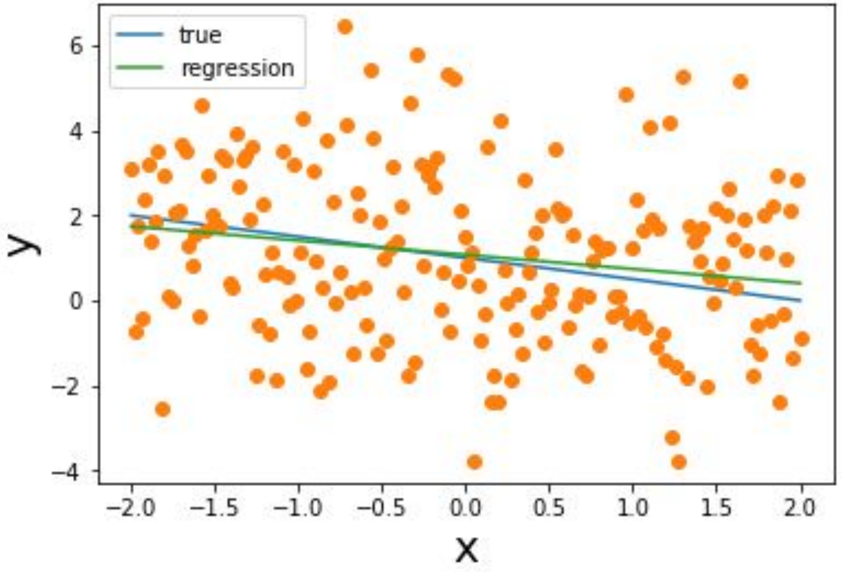
$$p(\mathbf{Y},\theta) = p(\theta,\mathbf{Y}) = p(\theta|\mathbf{Y})p(\mathbf{Y})$$

$$p(\mathbf{Y}|\theta)p(\theta) = p(\theta|\mathbf{Y})p(\mathbf{Y})$$

$$p(\theta|\mathbf{Y}) = \frac{p(\mathbf{Y}|\theta)p(\theta)}{p(\mathbf{Y})}$$



The posterior - means the probability distribution of model parameters given data. For example what is the probability that w=0 and b=0 given data points.



Understanding Bayesian approach

The marginal distribution normalizes the posterior. Since this is just a constant we can neglect this term for a moment:

$$p(\theta|\mathbf{Y}) = \frac{p(\mathbf{Y}|\theta)p(\theta)}{p(\mathbf{Y})} \longrightarrow p(\theta|\mathbf{Y}) \propto p(\mathbf{Y}|\theta)p(\theta)$$

The prior

$$p(\mathbf{Y}|\theta) = C^n \exp\left(-\frac{1}{2\sigma^2}\left(\mathbf{Y} - f(\mathbf{X})\right)^T\left(\mathbf{Y} - f(\mathbf{X})\right)\right) \qquad p(\theta) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(w^2 + b^2\right)\right)$$

Thus we have (neglecting constant terms)

$$p(\theta|\mathbf{Y}) \propto \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{N}\left(y_i - f(x_i)\right)^2\right)\exp\left(-\frac{1}{2}\left(w^2 + b^2\right)\right)$$

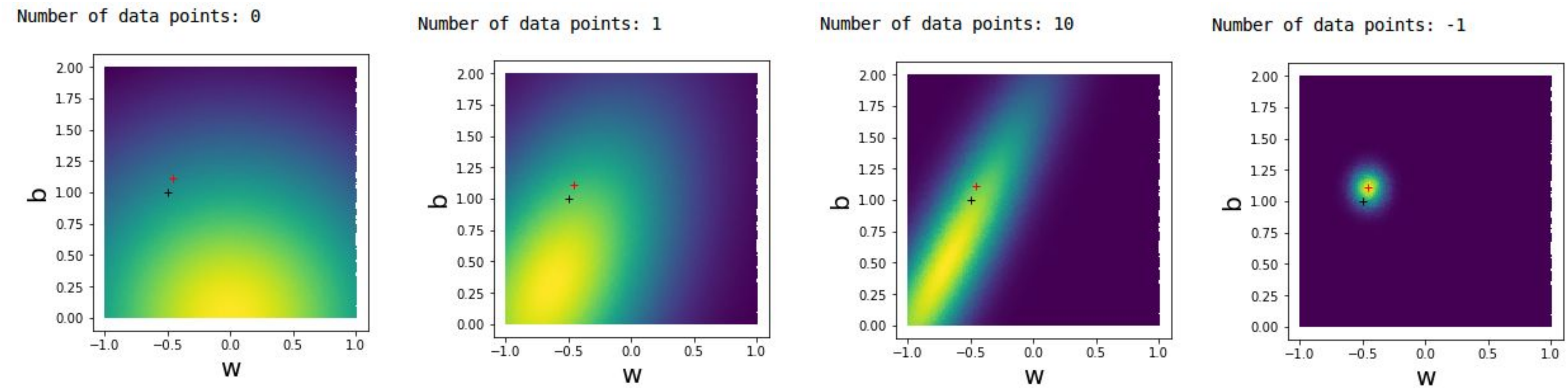
Note, when **N=0 (no data)** our posterior distribution becomes the **prior** one.

Understanding Bayesian approach - demo

The convergence of not normalized posterior in function of number of samples

$$p(\theta|\mathbf{Y}) \propto \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - f(x_i))^2\right)\exp\left(-\frac{1}{2}\left(w^2+b^2\right)\right)$$

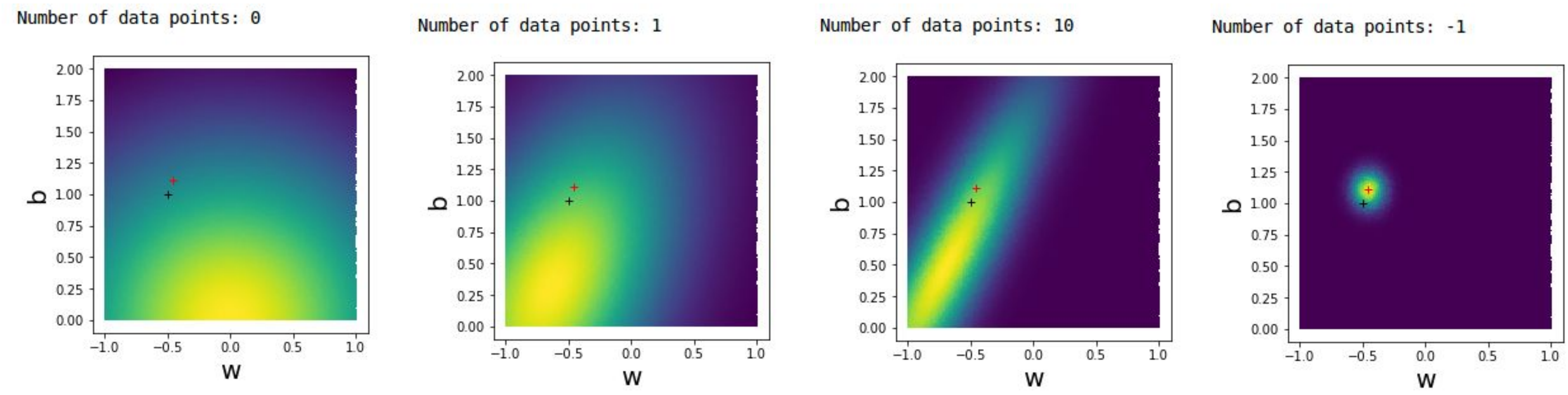
We can plot p(...) as a function of two parameters w and b and changing the number of N



black cross - true values, **red** obtained from MLE.

Understanding Bayesian approach - demo

$$p(\theta|\mathbf{Y}) \propto \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - f(x_i))^2\right)\exp\left(-\frac{1}{2}(w^2+b^2)\right)$$



Some observations:

- With more data, the posterior gets localized around true values of w and b and the impact of the prior distribution get lower and lower.
- Sometimes, even if we have small number of samples, the posterior may get localized around true values. However the uncertainty will be large enough.

The necessity of approximations

Our problem is simple enough and we can try to compute full posterior distribution analytically.

$$p(\theta|\mathbf{Y}) = \frac{p(\mathbf{Y}|\theta)p(\theta)}{p(\mathbf{Y})}$$

$$p(\theta|\mathbf{Y}) = \frac{\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{N}\left(y_i - f(x_i)\right)^2\right)\exp\left(-\frac{1}{2}\left(w^2+b^2\right)\right)}{\int dw db \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{N}\left(y_i - f(x_i)\right)^2\right)\exp\left(-\frac{1}{2}\left(w^2+b^2\right)\right)}$$

Even for this simple problem the integral in the denominator is not the easiest one to calculate… but it can be computed.

Consider the case of multivariate linear regression problem:

$$f(\mathbf{X}) = \mathbf{WX} + \mathbf{b}$$

The necessity of approximations

A multivariate linear regression from wikipedia. **Note** the lack of normalization constant

Posterior distribution [edit]

Using the above prior and likelihood, the posterior distribution can be expressed as[1]:

$$\begin{aligned}\rho(\boldsymbol\beta,\boldsymbol\Sigma_{\epsilon}|\mathbf{Y},\mathbf{X}) \propto\, & |\boldsymbol\Sigma_{\epsilon}|^{-(\nu_0+m+1)/2}\exp{(-\tfrac{1}{2}\operatorname{tr}(\mathbf{V_0}\boldsymbol\Sigma_{\epsilon}^{-1}))} \\ & \times|\boldsymbol\Sigma_{\epsilon}|^{-k/2}\exp{(-\tfrac{1}{2}\operatorname{tr}((\mathbf{B}-\mathbf{B_0})^{\rm T}\boldsymbol\Lambda_0(\mathbf{B}-\mathbf{B_0})\boldsymbol\Sigma_{\epsilon}^{-1}))} \\ & \times|\boldsymbol\Sigma_{\epsilon}|^{-n/2}\exp{(-\tfrac{1}{2}\operatorname{tr}((\mathbf{Y}-\mathbf{XB})^{\rm T}(\mathbf{Y}-\mathbf{XB})\boldsymbol\Sigma_{\epsilon}^{-1}))},\end{aligned}$$

where $\operatorname{vec}(\mathbf{B_0}) = \boldsymbol\beta_0$. The terms involving $\mathbf{B}$ can be grouped (with $\boldsymbol\Lambda_0 = \mathbf{U}^{\rm T}\mathbf{U}$) using:

$$\begin{aligned}& (\mathbf{B}-\mathbf{B_0})^{\rm T}\boldsymbol\Lambda_0(\mathbf{B}-\mathbf{B_0})+(\mathbf{Y}-\mathbf{XB})^{\rm T}(\mathbf{Y}-\mathbf{XB}) \\ & = \left(\begin{bmatrix}\mathbf{Y}\\ \mathbf{UB_0}\end{bmatrix}-\begin{bmatrix}\mathbf{X}\\ \mathbf{U}\end{bmatrix}\mathbf{B}\right)^{\rm T}\left(\begin{bmatrix}\mathbf{Y}\\ \mathbf{UB_0}\end{bmatrix}-\begin{bmatrix}\mathbf{X}\\ \mathbf{U}\end{bmatrix}\mathbf{B}\right) \\ & = \left(\begin{bmatrix}\mathbf{Y}\\ \mathbf{UB_0}\end{bmatrix}-\begin{bmatrix}\mathbf{X}\\ \mathbf{U}\end{bmatrix}\mathbf{B_n}\right)^{\rm T}\left(\begin{bmatrix}\mathbf{Y}\\ \mathbf{UB_0}\end{bmatrix}-\begin{bmatrix}\mathbf{X}\\ \mathbf{U}\end{bmatrix}\mathbf{B_n}\right)+(\mathbf{B}-\mathbf{B_n})^{\rm T}(\mathbf{X}^{\rm T}\mathbf{X}+\boldsymbol\Lambda_0)(\mathbf{B}-\mathbf{B_n}) \\ & = (\mathbf{Y}-\mathbf{XB_n})^{\rm T}(\mathbf{Y}-\mathbf{XB_n})+(\mathbf{B_0}-\mathbf{B_n})^{\rm T}\boldsymbol\Lambda_0(\mathbf{B_0}-\mathbf{B_n})+(\mathbf{B}-\mathbf{B_n})^{\rm T}(\mathbf{X}^{\rm T}\mathbf{X}+\boldsymbol\Lambda_0)(\mathbf{B}-\mathbf{B_n}),\end{aligned}$$

with

$$\mathbf{B_n}=(\mathbf{X}^{\rm T}\mathbf{X}+\boldsymbol\Lambda_0)^{-1}(\mathbf{X}^{\rm T}\mathbf{X}\hat{\mathbf{B}}+\boldsymbol\Lambda_0\mathbf{B_0})=(\mathbf{X}^{\rm T}\mathbf{X}+\boldsymbol\Lambda_0)^{-1}(\mathbf{X}^{\rm T}\mathbf{Y}+\boldsymbol\Lambda_0\mathbf{B_0}).$$

This now allows us to write the posterior in a more useful form:

$$\begin{aligned}\rho(\boldsymbol\beta,\boldsymbol\Sigma_{\epsilon}|\mathbf{Y},\mathbf{X}) \propto\, & |\boldsymbol\Sigma_{\epsilon}|^{-(\nu_0+m+n+1)/2}\exp{(-\tfrac{1}{2}\operatorname{tr}((\mathbf{V_0}+(\mathbf{Y}-\mathbf{XB_n})^{\rm T}(\mathbf{Y}-\mathbf{XB_n})+(\mathbf{B_n}-\mathbf{B_0})^{\rm T}\boldsymbol\Lambda_0(\mathbf{B_n}-\mathbf{B_0}))\boldsymbol\Sigma_{\epsilon}^{-1}))} \\ & \times|\boldsymbol\Sigma_{\epsilon}|^{-k/2}\exp{(-\tfrac{1}{2}\operatorname{tr}((\mathbf{B}-\mathbf{B_n})^{\rm T}\boldsymbol\Lambda_0)(\mathbf{B}-\mathbf{B_n})\boldsymbol\Sigma_{\epsilon}^{-1}))}.\end{aligned}$$

The necessity of approximations

The calculations of posterior become tedious even for simplest models, additionally only a small fraction of them has analytical solution e.g. it cannot be done for arbitrary deep neural networks.

Solution: we need a method which will allow us to approximate the posterior.

One of the approaches is

Variational Inference

Which converts the problem of computing posterior as optimization problem hence we can use SGD for it.

Note: Variational means infinitesimal change in independent variable or function

Variational Inference - derivation of the ELBO

The frequentist optimizes parameters by doing MLE

$$p(\mathbf{Y}|\theta) = C^n \exp\left(-\frac{1}{2\sigma^2}(\mathbf{Y} - f(\mathbf{X}))^T(\mathbf{Y} - f(\mathbf{X}))\right)$$

$$\mathcal{L}_\theta = \log(p) = \log C^n - \frac{1}{2\sigma^2}(\mathbf{Y} - f(\mathbf{X}))^T(\mathbf{Y} - f(\mathbf{X}))$$

We want to find similar method which will allow us to do the SGD but with additional information about posterior distribution.

We know that computing **exact posterior** is not feasible in general, hence we propose a variational distribution **Q**

$$p(\theta|\mathbf{Y}) = \frac{p(\mathbf{Y}|\theta)p(\theta)}{p(\mathbf{Y})}$$

$$Q^*(\theta) = \underset{\theta}{\text{argmin}} KL\left(Q(\theta)||p(\theta|\mathbf{Y})\right)$$

Q is an arbitrary distribution which in general is differentiable and easy to sample from

Variational Inference - derivation of the ELBO

We are going to simply things in order to make our problem solvable

Bayes rule

$$p(\theta|\mathbf{Y}) = \frac{p(\mathbf{Y}|\theta)p(\theta)}{p(\mathbf{Y})}$$

$$Q^*(\theta) = \underset{\theta}{\operatorname{argmin}} KL\left(Q(\theta)||p(\theta|\mathbf{Y})\right)$$

$$KL\left(Q(\theta)||p(\theta|\mathbf{Y})\right) = \int d\theta Q(\theta)\log\left(\frac{Q(\theta)}{p(\theta|\mathbf{Y})}\right)$$

$$= \boxed{\int d\theta Q(\theta)\log Q(\theta) - \int d\theta Q(\theta)\log p(\theta)} - \int d\theta Q(\theta)\log p(\mathbf{Y}|\theta) + \boxed{\int d\theta Q(\theta)\log p(\mathbf{Y})}$$

This can be further simplified:

$$= \underbrace{KL\left(Q(\theta)||p(\theta)\right) - \mathbb{E}_{\theta\sim Q(\theta)}\log p(\mathbf{Y}|\theta)}_{\text{-ELBO}} + \log p(\mathbf{Y})$$

Variational Inference - derivation of the ELBO

By definition ELBO is equal to

$$ELBO = \mathbb{E}_{\theta \sim Q(\theta)} \log p(\mathbf{Y}|\theta) - KL\left(Q(\theta)||p(\theta)\right)$$

From previous slide we have:

$$KL\left(Q(\theta)||p(\theta|\mathbf{Y})\right) = -ELBO + \log p(\mathbf{Y})$$

$$KL\left(Q(\theta)||p(\theta|\mathbf{Y})\right) + ELBO = \log p(\mathbf{Y})$$

$$\text{positive(intractable)} + \text{ELBO(tractable)} = \text{const}$$

The evidence is const w.r.t variational parameters and KL is nonnegative, hence ELBO is a lower bound for evidence: ELBO - Evidence Lower BOund.

Solution: In order to minimize KL it is enough to maximize the second term i.e. ELBO.

$$ELBO \leq \log p(\mathbf{Y})$$

Variational Inference

We want to optimize the model parameters by maximizing ELBO

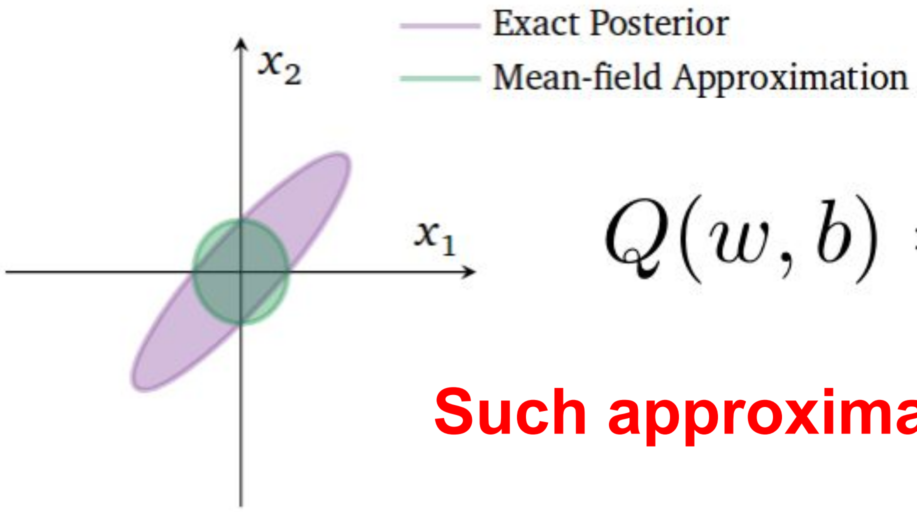
$$ELBO = \mathbb{E}_{\theta \sim Q(\theta)} \log p(\mathbf{Y}|\theta) - KL\left(Q(\theta)||p(\theta)\right)$$

likelihood

we want to keep variational posterior close to the prior

Let's return to our original problem: y=wx+b

The simplest possible choice for Q is to assume that all parameters are independent:



$$Q(w,b) = \frac{1}{2\pi\sigma_w\sigma_b} e^{-\frac{1}{2\sigma_w^2}(w-\mu_w)^2} e^{-\frac{1}{2\sigma_b^2}(b-\mu_b)^2}$$

Such approximation for Q is called Mean Field Approximation

Variational Inference

We want to optimize the model parameters by maximizing ELBO

$$ELBO = \mathbb{E}_{\theta\sim Q(\theta)} \log p(\mathbf{Y}|\theta) - KL(Q(\theta)||p(\theta))$$

What we have so far:

- variational distribution: $$Q(w,b) = \frac{1}{2\pi\sigma_w\sigma_b} e^{-\frac{1}{2\sigma_w^2}(w-\mu_w)^2} e^{-\frac{1}{2\sigma_b^2}(b-\mu_b)^2}$$

- the likelihood: $$p(\mathbf{Y}|\theta) = C^n \exp\left(-\frac{1}{2\sigma^2}(\mathbf{Y} - f(\mathbf{X}))^T(\mathbf{Y} - f(\mathbf{X}))\right)$$

- the expectation of likelihood up to a constant term

$$\mathbb{E}_{\theta\sim Q(\theta)} \log p(\mathbf{Y}|\theta) \propto -\frac{1}{2\sigma^2}\sum_{i=1}^{N}\int dwdbQ(w,b)\left(y_i - f(x_i;w,b)\right)^2$$

- KL divergence - note it does not depend on data

$$KL(Q(\theta)||p(\theta)) = \int dwdbQ(w,b)\log\left(\frac{Q(w,b)}{p(w,b)}\right)$$

Variational Inference - exact solution

We want to optimize the model parameters by maximizing ELBO

$$ELBO = \mathbb{E}_{\theta \sim Q(\theta)} \log p(\mathbf{Y}|\theta) - KL\left(Q(\theta)||p(\theta)\right)$$

All the integral can be computed analytically:

$$\begin{aligned} ELBO &= -\frac{1}{2\sigma^2}\left\{S_{y^2} - 2\mu_w S_{xy} - 2\mu_b S_y + \left(\mu_w^2 + \sigma_w^2\right) S_{x^2} + \right. \\ &+ \left. 2\mu_b\mu_w S_x + \left(\mu_b^2 + \sigma_b^2\right) S_1\right\} - \frac{\mu_w^2 + \sigma_w^2 + \mu_b^2 + \sigma_b^2}{2} \\ &+ \left\{\log\sigma_b + \log\sigma_w + 1\right\} \end{aligned}$$

Where S_P is the sum over P: $S_{x^2} = \sum_i^N x_i^2$

Variational Inference - exact solution - demo

We want to optimize the model parameters by maximizing ELBO

$$ELBO = \mathbb{E}_{\theta \sim Q(\theta)} \log p(\mathbf{Y}|\theta) - KL\left(Q(\theta)||p(\theta)\right)$$

The next step is to find maximum value for ELBO by finding optimal set of parameters:

$$\frac{\partial ELBO}{\partial \mu_w} = 0 \quad = \quad -\frac{1}{2\sigma^2}\{-2S_{xy} + 2\mu_w S_{x^2} + 2\mu_b S_x\} - \mu_w$$
$$\frac{\partial ELBO}{\partial \mu_b} = 0 \quad = \quad -\frac{1}{2\sigma^2}\{-2S_y + 2\mu_w S_x + 2\mu_b S_1\} - \mu_b$$
$$\frac{\partial ELBO}{\partial \sigma_w} = 0 \quad = \quad -\frac{\sigma_w S_{x^2}}{\sigma^2} - \sigma_w + \frac{1}{\sigma_w}$$
$$\frac{\partial ELBO}{\partial \sigma_b} = 0 \quad = \quad -\frac{\sigma_b S_1}{\sigma^2} - \sigma_b + \frac{1}{\sigma_b}$$

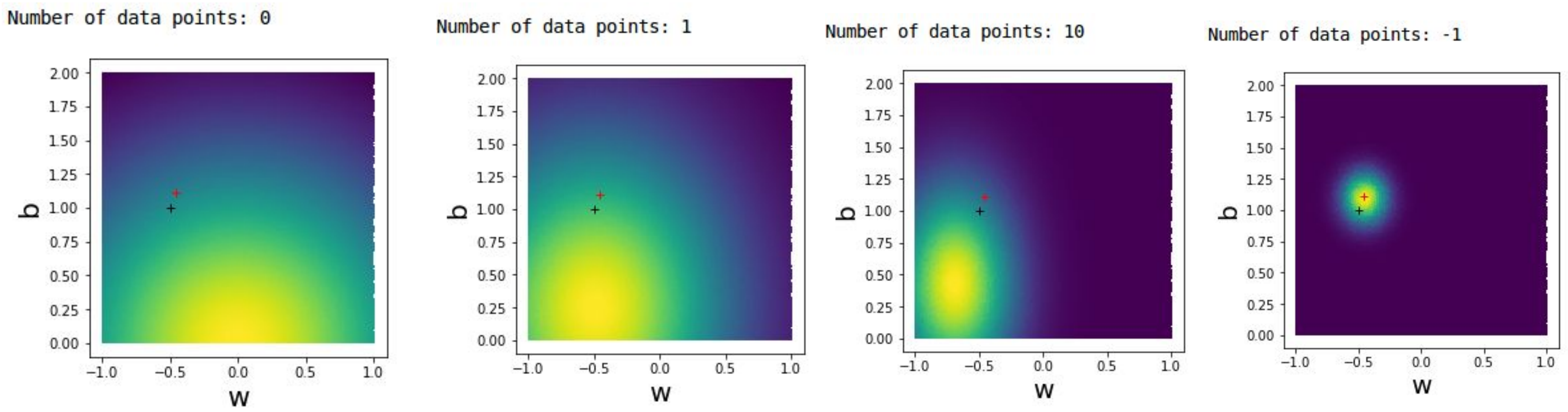
This system of equation can be easily solved for optimal: $\mu_w, \mu_b, \sigma_w, \sigma_b$

Variational Inference - exact solution - demo

The analytical solution can be compared with previous approach

```python
def mean_field_posterior(y, x, theta):
    w, b = theta
    sx = x.sum()
    sy = y.sum()
    sxy = (x*y).sum()
    sxx = (x*x).sum()
    s = 2
    s1 = sum([1]*len(x))

    sw = 1/np.sqrt(sxx/s**2 + 1)
    sb = 1/np.sqrt(s1/s**2 + 1)
    # generated code by maxima
    mw = ((s**2+s1)*sxy-sx*sy)/(s**4+s**2*s1-sx**2+(s1+s**2)*sxx)
    mb = -((-sxx*sy)-s**2*sy+sx*sxy)/(s**4+s**2*s1-sx**2+(s1+s**2)*sxx)

    return 1/(2*np.pi*sw*sb)*np.exp(-0.5/(sw**2)*(w - mw)**2)*np.exp(-0.5/(sb**2)*(b - mb)**2)
```

$$Q(w,b) = \frac{1}{2\pi\sigma_w\sigma_b} e^{-\frac{1}{2\sigma_w^2}(w-\mu_w)^2} e^{-\frac{1}{2\sigma_b^2}(b-\mu_b)^2}$$



Variational Inference - iterative approach

Usually we cannot compute integrals analytically and obtain this nice expressions for gradients:

$$
\begin{aligned}
\frac{\partial ELBO}{\partial \mu_w} = 0 \quad &= \quad -\frac{1}{2\sigma^2}\{-2S_{xy} + 2\mu_w S_{x^2} + 2\mu_b S_x\} - \mu_w \\
\frac{\partial ELBO}{\partial \mu_b} = 0 \quad &= \quad -\frac{1}{2\sigma^2}\{-2S_y + 2\mu_w S_x + 2\mu_b S_1\} - \mu_b \\
\frac{\partial ELBO}{\partial \sigma_w} = 0 \quad &= \quad -\frac{\sigma_w S_{x^2}}{\sigma^2} - \sigma_w + \frac{1}{\sigma_w} \\
\frac{\partial ELBO}{\partial \sigma_b} = 0 \quad &= \quad -\frac{\sigma_b S_1}{\sigma^2} - \sigma_b + \frac{1}{\sigma_b}
\end{aligned}
$$

Note that the expressions above were derived from evaluation of integrals and then by taking gradient of ELBO w.r.t approximated posterior parameters

$$
\mathbb{E}_{\theta \sim Q(\theta)} \log p(\mathbf{Y}|\theta) \propto -\frac{1}{2\sigma^2}\sum_{i=1}^{N}\int dwdbQ(w,b)\left(y_i - f(x_i; w, b)\right)^2
$$

$$
KL\left(Q(\theta)||p(\theta)\right) = \int dwdbQ(w,b)\log\left(\frac{Q(w,b)}{p(w,b)}\right)
$$

Variational Inference - iterative approach

In general we approximate ELBO by sampling from posteriors Q

$$ELBO = \mathbb{E}_{\theta \sim Q(\theta)} \log p(\mathbf{Y}|\theta) - KL\left(Q(\theta)||p(\theta)\right)$$

$$ELBO \quad = \boxed{-\frac{1}{2\sigma^2} \sum_{i=1}^{N} \frac{1}{N_s} \sum_{w_k, b_k \sim Q}^{N_s} (y_i - f(x_i; w_k, b_k))^2}$$
$$\boxed{- \quad \frac{\mu_w^2 + \sigma_w^2 + \mu_b^2 + \sigma_b^2}{2} + \{\log \sigma_b + \log \sigma_w + 1\}}$$

If Q and p are standard distributions like gaussian, there are exact formulas for computing KL(Q||p)

Then we compute gradients of ELBO and perform parameter update using SGD

Variational Inference - iterative approach

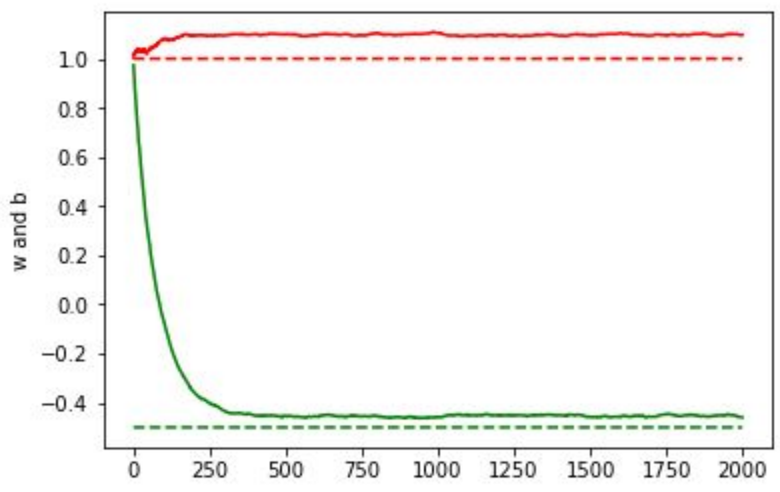
However for certain distributions we can compute KL term

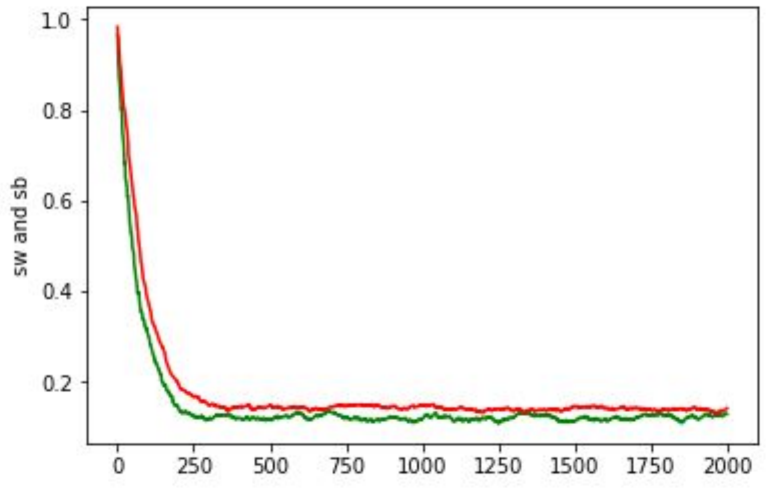
$$W = W_{mean} + \epsilon W_{sigma}$$

$$\epsilon \sim N(0,1)$$

$$ELBO = -\frac{1}{2\sigma^2}\sum_{i=1}^{N}\frac{1}{N_s}\sum_{w_k,b_k\sim Q}^{N_s}(y_i - f(x_i;w_k,b_k))^2 - \frac{\mu_w^2+\sigma_w^2+\mu_b^2+\sigma_b^2}{2} + \{\log\sigma_b + \log\sigma_w + 1\}$$

Naive implementation of gradient descent with sampling:





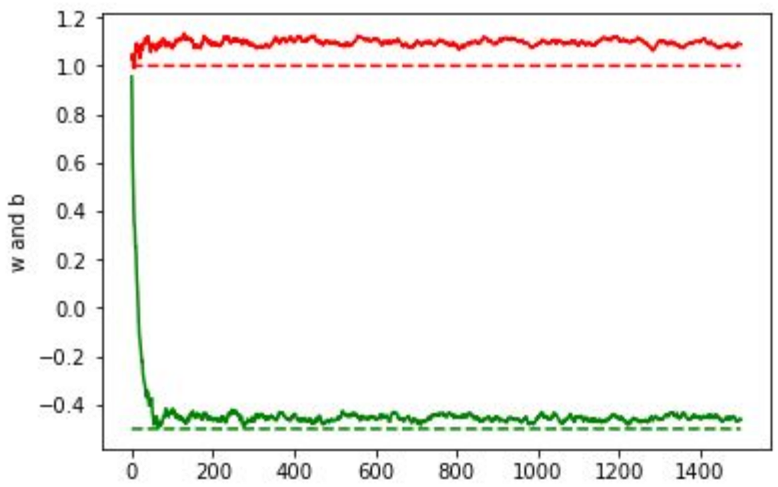

```
for i in range(Ni):
    a = -1/(sigma**2 * Ns)
    eps_w = np.random.randn(Ns)
    eps_b = np.random.randn(Ns)

    grad_mw = 0
    grad_mb = 0
    grad_sw = 0
    grad_sb = 0

    for k in range(Ns):
        wk = mw + eps_w[k] * sw
        bk = mb + eps_b[k] * sb

        Dki = y - f(x, wk, bk)

        grad_mw += np.sum(Dki * x)
        grad_sw += np.sum(Dki * x) * eps_w[k]

        grad_mb += np.sum(Dki)
        grad_sb += np.sum(Dki) * eps_b[k]

    grad_mw = a * grad_mw + mw
    grad_sw = a * grad_sw + sw - 1/sw

    grad_mb = a * grad_mb + mb
    grad_sb = a * grad_sb + sb - 1/sb

    mw = mw - lr * grad_mw
    mb = mb - lr * grad_mb
    sw = sw - lr * grad_sw
    sb = sb - lr * grad_sb
```

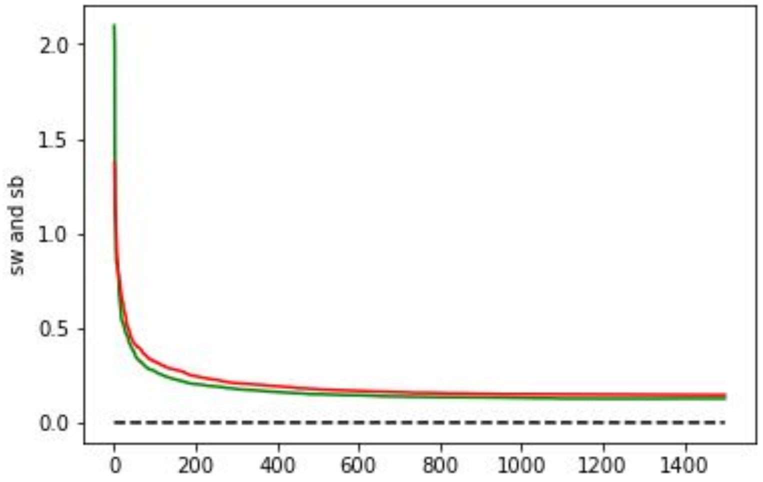
Variational Inference - iterative approach

For more stable training replace sigma with following parametrization:

$$\sigma = \exp(\sigma')$$

This keeps sigma to be always positive.



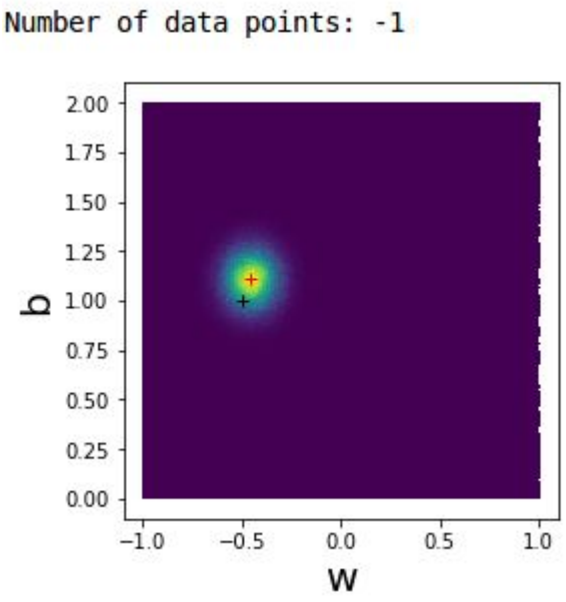


Note that sigmas became more stable during training process

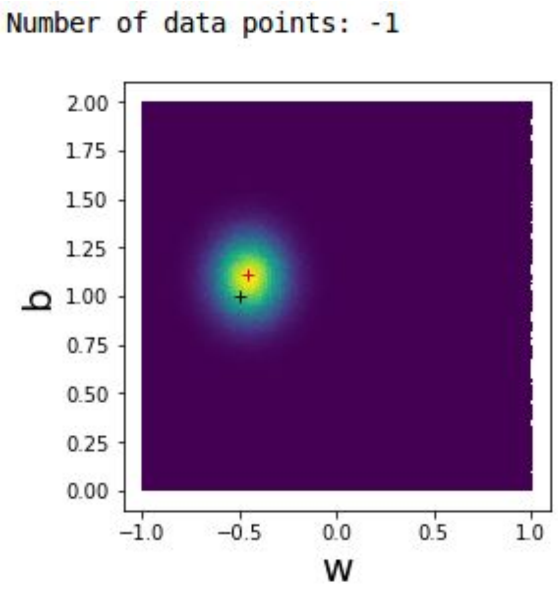
```python
for i in range(Ni):
    a = -1/(sigma**2 * Ns)
    eps_w = np.random.randn(Ns)
    eps_b = np.random.randn(Ns)

    grad_mw = 0
    grad_mb = 0
    grad_pw = 0
    grad_pb = 0
    lh_loss = 0
    for k in range(Ns):
        wk = mw + eps_w[k] * np.exp(pw)
        bk = mb + eps_b[k] * np.exp(pb)

        Dki = y - f(x, wk, bk)

        grad_mw += np.sum(Dki * x)
        grad_pw += np.sum(Dki * x) * eps_w[k] * np.exp(pw)

        grad_mb += np.sum(Dki)
        grad_pb += np.sum(Dki) * eps_b[k] * np.exp(pb)

        lh_loss += np.mean(Dki**2)

    grad_mw = a * grad_mw + mw
    grad_pw = a * grad_pw + np.exp(2*pw) - 1

    grad_mb = a * grad_mb + mb
    grad_pb = a * grad_pb + np.exp(2*pb) - 1

    mw = mw - lr * grad_mw
    mb = mb - lr * grad_mb
    pw = pw - lr * grad_pw
    pb = pb - lr * grad_pb
```

Variational Inference - comparison

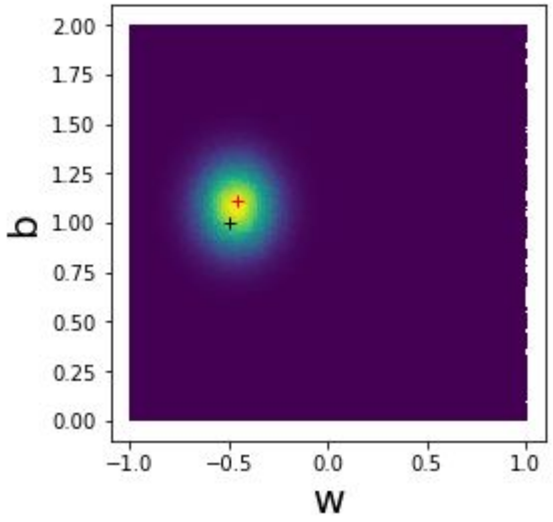
Finally we can compare the results from all approaches:



Unnormalized exact posterior



Exact variational posterior



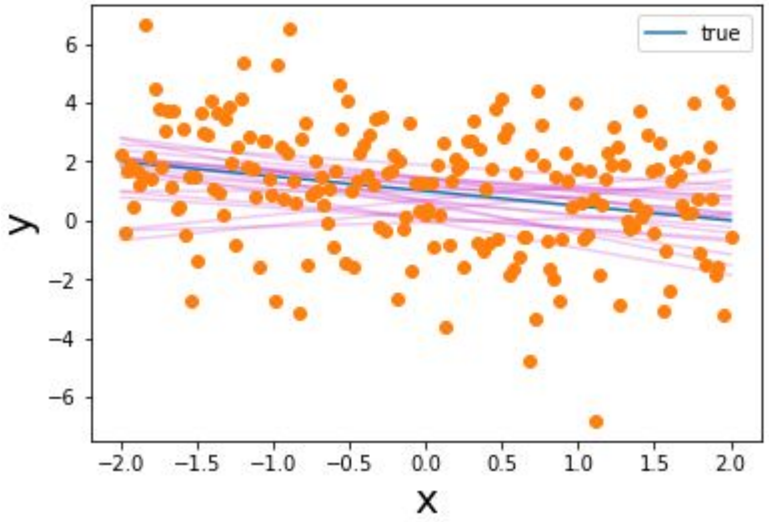
Iterative variational posterior

```
def sgd_mean_field_posterior(theta):
    w, b = theta
    sw = np.exp(pw)
    sb = np.exp(pb)
    return 1/(2*np.pi*sw*sb) \
            * np.exp(-0.5/(sw**2)*(w - mw)**2)\
            * np.exp(-0.5/(sb**2)*(b - mb)**2)
```

Variational Inference - sampling

After convergence we can sample our network in order to perform prediction

```
def sample_wb():
    wk = mw + np.random.randn() * np.exp(pw)
    bk = mb + np.random.randn() * np.exp(pb)
    return wk, bk

for s in range(20):
    wk, bk = sample_wb()
    fs = f(x, wk, bk)
    plt.plot(x, fs, 'm-', alpha=0.2)
```



General approach

- Define model, priors and posteriors
- Train the model by optimizing ELBO, set number of MC samples

$$ELBO = \mathbb{E}_{\theta \sim Q(\theta)} \log p(\mathbf{Y}|\theta) - KL\left(Q(\theta)||p(\theta)\right)$$

- Sample model weights from trained posteriors and predict

Variational Inference - in Edward - demos

We want to solve the same problem with Edward library

Edward

http://edwardlib.org/tutorials/

General approach

- Define model, priors and posteriors
- Train the model by optimizing ELBO, set number of MC samples

$$ELBO = \mathbb{E}_{\theta \sim Q(\theta)} \log p(\mathbf{Y}|\theta) - KL\left(Q(\theta)||p(\theta)\right)$$

- Sample model weights from trained posteriors and predict

```
from edward.models import Normal
import edward as ed

X = tf.placeholder(tf.float32, [N, D])
w = Normal(loc=tf.zeros(D), scale=tf.ones(D))
y = Normal(loc=ed.dot(X, w), scale=tf.ones(N))
```

```
qw = Normal(loc=tf.Variable(tf.random_normal([D])),
            scale=tf.nn.softplus(tf.Variable(tf.random_normal([D]))))
```

```
inference = ed.KLqp({w: qw}, data={X: x_train, y: y_train})
inference.run(n_samples=15, n_iter=250)
```

```
def visualise(X_data, y_data, w, n_samples=10):
    w_samples = qw.sample(n_samples).eval()
    plt.scatter(X_data[:, 0], y_data)
    for ns in range(n_samples):
        output = np.dot(X_data, w_samples[ns])
        plt.plot(X_data[:, 0], output, 'r-', alpha=0.4)
```

Questions?

References

- https://arxiv.org/pdf/1601.00670.pdf - A review on VI
- https://pl.wikipedia.org/wiki/Wnioskowanie_bayesowskie - polish naming for Bayesian statistics

FORNAX

WWW.FORNAX.AI